Project 2: Advanced Wireless Protocol Security Analysis (Zigbee/BLE)

Implementation Roadmap & Execution Guide

Project Duration: 2-4 months | **Complexity:** High | **Target Companies:** NVIDIA, Intel, AMD, Samsung, Apple
Publication Potential: YES | **Resume Impact:** HIGH

EXECUTIVE SUMMARY

This document provides a complete implementation guide for **Project 2: Advanced Wireless Protocol Security Analysis**, focusing on Zigbee and Bluetooth Low Energy (BLE) protocol vulnerability identification and exploitation. The project leverages existing open-source and commercial tools while developing custom modules for advanced traffic analysis, protocol fuzzing, and security assessment.

Expected Outcomes:

- Unified analysis framework combining Zigbee + BLE security testing
- Custom packet generation and manipulation modules
- Automated vulnerability detection and reporting
- Publication-ready research paper
- Open-source toolkit for community use

PART 1: EXISTING SOLUTIONS ANALYSIS

1.1 Competitive Landscape Matrix

Tool Name	Type	Primary Focus	Key Advantages	Major Drawbacks	Limitations	Cost	License
Ubertooth One	Hardware/OS	BLE Packet Sniffing	Industry standard, full BLE stack analysis, Wireshark integration, real-time sniffing	Requires dedicated hardware ($120), limited to 2.4 GHz, slower connection following	Cannot sniff encrypted payloads without keys, lacks Zigbee support	$100-150	Open-source (GPLv2)
Nordic nRF Sniffer	Hardware/Windows	BLE Monitoring	Easy Windows integration, vendor support, good for development	Windows-only, limited to Nordic devices, obsolete Wireshark version (1.12)	Outdated software stack, poor cross-platform support	$50-60	Commercial
ApiMote v4	Hardware/Open	802.15.4/Zigbee	Purpose-built for Zigbee, open-source hardware, KillerBee compatibility, high-speed capturing	Requires hardware assembly or pre-programmed purchase, steep learning curve	Limited to single-channel monitoring, not portable	$60-100	Open-source (Hardware)

Tool Name	Type	Primary Focus	Key Advantages	Major Drawbacks	Limitations	Cost	License
KillerBee	Software/ CLI	Zigbee Assessment	Comprehensive packet crafting, APS/ZCL layer support, scapy-based architecture	Poor documentation, unmaintained active development, dependency issues	Lacks modern web UI, fragile Wireshark integration	Free	Open-source (GPLv2)
Wireshark (802.15.4)	Software/ GUI	Protocol Analysis	Industry-standard dissector, universal format support, packet visualization	Passive analysis only (requires sniffer hardware), steep UI learning curve for custom dissection	Cannot generate packets, limited scripting automation	Free	Open-source (GPLv2)
Zigator	Software/ CLI	Encrypted Zigbee Analysis	Detects jamming/spoofing in encrypted packets, traffic classification, device fingerprinting	Limited to pre-processed PCAPs, no real-time capturing, minimal exploit	Passive-only tool, not designed for active testing	Free	Open-source (MIT)

Tool Name	Type	Primary Focus	Key Advantages	Major Drawbacks	Limitations	Cost	License
				generation			
Z-Fuzzer	Software/CLI	Zigbee Fuzzing	Coverage-guided mutation testing, discovers 0-days (CVE-validated), higher code coverage vs BooFuzz/Peach	Requires Zigbee simulator setup (Z-Stack), slow testing cycles, false positives in mutation	Limited to Z-Stack implementation, not vendor-agnostic	Free	Open-source (MIT)
Scapy	Library/Python	Packet Crafting	Universal protocol support, layer-by-layer control, extensible architecture, excellent documentation	Requires Python development, no GUI, significant code complexity for complex protocols	Steep learning curve, performance bottlenecks at scale	Free	Open-source (GPLv2)
RFQuack	Hardware/CLI	RF Protocol Analysis	Flexible RF hardware interface, firmware-based	Requires SDR hardware knowledge,	Requires external RF hardware,	Free	Open-source (MIT)

Tool Name	Type	Primary Focus	Key Advantages	Major Drawbacks	Limitations	Cost	License
			filtering, multi-radio support, interactive shell	limited Zigbee/BLE-specific dissectors	not plug-and-play		
BLE Security Framework (Custom)	Software/Python	BLE Replay/Hijacking	Application-layer encryption detection (BLECryptracer), MITM attack simulation	Static analysis only (Android APKs), not runtime dynamic testing	Limited to mobile BLE implementations	Free	Research-only
BlueSWAT	Software/Rust	BLE State-Machine Monitoring	Session-level attack detection, FSM-based monitoring, anti-replay mechanisms	Focused on defense only (not offensive research), limited exploit generation	Not suitable for active penetration testing	Free	Open-source (Research)
Spanalytics PANalyzr	Hardware/Comme	Full ISM Band Analysis	Multi-protocol simultaneous analysis (BLE+Bluetooth+Zigb	Extremely high cost ($15K-40K), closed-source,	Enterprise-only pricing, not accessible for	$15K-40	Commercial (Propr

Tool Name	Type	Primary Focus	Key Advantages	Major Drawbacks	Limitations	Cost	License
	rcial		ee), high-end professional sniffer	vendor lock-in	research	K/yr	ietary)
Packet Squirrel/ Offensive IoT	Commercial Training	Offensive IoT Testing	Pre-configured hardware, comprehensive training, BLE/Zigbee labs included	Expensive for educational purpose ($5K+), vendor-specific tools	Limited to training environment, not production-grade	$3K-7K	Commercial (Training)
ATLAS Framework	Software/ Open	BLE Localization	Anonymous tracking using BLE, privacy-preserving design, open-source	Focused on localization (not security), limited exploit modules	Not designed for security assessment	Free	Open-source (Apache 2.0)
Bluetooth Exploitation Tools (Misc)	Mixed	BLE Specific Attacks	BlueMaho (multi-exploit), Bluetrap (comprehensive framework), BTCrawler	Fragmented ecosystem, variable maintenance, community-driven quality	Many abandoned projects, inconsistent documentation	Free	Open-source (Variable)

Tool Name	Type	Primary Focus	Key Advantages	Major Drawbacks	Limitations	Cost	License
			(reconnaissance)				

1.2 Open-Source vs Commercial Analysis

Dimension	Open-Source Solutions	Commercial Solutions
Cost	$0-150 (hardware only)	$15K-40K/year
Development Model	Community-driven, async updates	Vendor-managed, regular patches
Customization	Unlimited source code access	Black-box, limited APIs
Documentation	Variable quality, fragmented	Professional, vendor-supported
Protocol Coverage	BLE-strong, Zigbee-moderate, LoRa-weak	Full ISM band (BLE+Zigbee+WiFi+proprietary)
Active Development	Maintenance-mode (KillerBee), emerging (RFQuack)	Continuous innovation
Hardware Flexibility	High (multiple platforms)	Low (proprietary devices)
Suitable For	Researchers, red teams, hobby projects	Enterprise CISO programs, consulting firms

Recommendation: Combine **Ubertooth** + **ApiMote** + **KillerBee** + **Scapy** as foundation; build custom modules for advanced fuzzing and reporting.

PART 2: CUSTOM MODULE ARCHITECTURE

2.1 Module Specifications & Work Breakdown

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
MOD-1	**Unified Packet Sniffer**	Capture BLE/Zigbee packets from multiple hardware sources (Ubertooth, ApiMote, nRF51) and normalize into single PCA	BLE (all channels), Zigbee (all 16 channels), packet filtering, real-time processing, storage >1GB/hour	No packet injection yet; read-only mode; legal sniffing in authorized networks only	Hardware detection → driver initialization → packet capture loop → format conversion → storage/stream	Ubertooth + ApiMote hardware; Python 3.10+; pyusb, pcapy libraries	None (base module)	Unit: capture 10K packets/min; Integration: multi-device simultaneous capture; Stress: 8hr continuous capture withou	≥99.5% packet capture rate; <100ms latency between capture and storage; Support ≥3 concurrent hardw

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		P format						t packet loss	are sources
MOD-2	**Protocol Dissector Engine**	Decode BLE/Zigbee layer stack (PHY → MAC → NWK → APS → ZCL/ATT) and extract application-layer payl	BLE: LL/LE Meta/SM/ATT/GATT layers; Zigbee: PHY/MAC/NWK/APS/ZCL layers; unknown packet fallback	Must respect encrypted payloads (no forced decryption); decryption keys only when pro	Protocol spec parsing (IEEE 802.15.4 + Zigbee Alliance + BLE spec 5.4) → layer-wise decoding → cross-refe	BLE/Zigbee protocol specifications (PDF); vendor cluster definitions (XLS); Python dissector libri	MOD-1 (packet source); Wireshark dissector library (optional)	Unit: decode 1K packets/sec; Validation: compare against Wireshark output for 500+ packet typ	Parse 99%+ valid packets; ≤1% false-positive cluster identification; Support 200+ ZCL clus

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		oads; attribute mapping to device vendors		vided by user	rence vendor clusters → attribute database lookup	es (Scapy extensions)		es; Accuracy: ≥98% correct decoding	ters + 100 + GATT attributes
MOD-3	**Traffic Classification & Analysis**	Identify device types (sensor, light, switch, etc.), communication patterns	Device profiling (vendor/model detection); pattern analysis (sequence ident	Passive analysis only (no modification); pattern analysis based	Network traffic capture (MOD-1) → statistical analysis (packet timing,	CIC IoT Dataset 2022; device fingerprint DB; MITRE ATT&CK	MOD-1, MOD-2 (decoded packets); ML framework opt	Unit: classify 500 device instances; Integration: live classification on stre	Identify 90%+ of known device types; detect 95%+ comm uni

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		(periodic, event-driven), potential vulnerabilities (weak encryption, default keys, replay exposure)	ification); vulnerability scoring (CVSS-like model)	on published specs only	size, frequency) → cluster matching (ML-optional) → vulnerability mapping to known CVEs	device taxonomy; Python (pandas, numpy for analysis)	ional (scikit-learn)	aming traffic; ML validation: 10-fold cross-validation on labeled dataset	cation patterns; CVSS scoring ±0.5 error margin
MOD-4	**PacketForg**	Craft custom BLE/	Layer-wise packet	User must exp	Parse protocol spec	Scapy 2.5+; har	MOD-1, MOD	Unit: forge 100	Generate valid

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
	e & Replay Engine	Zigbee packets at any OSI layer; replay captured packets with modifications (field-level mutation) for active testing	construction; field-level parameter control; sequence-aware replay (e.g., follow authentication); frame counter handling	licitly authorize packet transmission (not automated DoS); generated packets logged with timestamps	→ build layer objects (Scapy) → assemble frame → calculate CRC/frame counter → send via hardware API	dware USB API; Zigbee/BLE frame counter trackers	-2 (for reference); Scapy packet templates	packets/sec; Integration: replay sequence of 1K+ packets with state tracking; Fuzzing: mutation-based gen	BLE/Zigbee frames (99%+ pass CRC); replay accuracy ±1ms timing; support ≥50 packet templates per protocol

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
								eration	
MOD-5	**Fuzzing & Mutation Engine**	Systematic fuzzing of Zigbee/BLE implementations using grammar-based and mutation-based approaches; disc	Coverage-guided fuzzing (AFL-style); field-specific mutation (XOR, bitflip, payload injection); state-aware fuzzing	Fuzzing limited to test environments (simulator or isolated lab setup); crash/hang logs kept confidentiall	Fuzz input generation (grammar/mutation) → packet crafting (MOD-4) → transmission → response capture (MOD-1) →	AFL/AFL++ (for coverage guidance), libFuzzer (optional); Z-Stack simulator (for Zigbee); test harness	MOD-1, MOD-4 (packet generation); protocol spec (grammar definition)	Unit: generate 10K test cases/hour; Integration: continuous fuzzing >100 hours; Crash triage: manual anal	Discover ≥5 new vulnerabilities; 95%+ of crashes triaged; code coverage >75% of target

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		over protocol violations and security flaws	(follow protocol state machine)	until vendor patch	crash analysis → triage	code (Python/C)		ysis of anomalies	implementation
MOD-6	**Key Extraction & Decryption**	Identify network/link keys from captured traffic, device firmware, or known defaults;	Support: ZigBee default key (ZigBeeAlliance09), manufacturer hard coded keys (by OUI matching)	Decryption only for traffic user has legal right to analyze; results quaran	Key database look up (by vendor/device MAC) → AES-CCM decryption (Zigbee) or AES-GCM	AES crypto library (pycryptodome); device firmware hash DB (for OUI ma	MOD-2 (packet decode); MOD-3 (device identification)	Unit: decrypt 500 packets/sec; Validation: test against known plaintext packets	Decrypt 99%+ of encrypted payloads where key available; identify 95

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		decrypt APS/application-layer payloads for plaintext analysis	, firmware extraction keys, user-provided keys	tined (no unauthorized data exfiltration)	(BLE) → payload extraction → format detection	ppi ng); Wireshark integration (optional)		; Accuracy: ≥100% correct decryption	%+ default/hardcoded keys from firmware
MOD-7	**Vulnerability Scanner & Mapper**	Detect security flaws in captured traffic: default keys, weak	CVE/CWE mapping; CVSS scoring; vulnerability severity classification; devi	Scanner is informational only (no active exploitation	Traffic analysis (MOD-1, MOD-2, MOD-3) → pattern matching	Vulnerability DB (CAN/CISA databases); CVSS calculat	MOD-1 through MOD-6 (data sources); Vulne	Unit: scan 10K packets; Integration: live scanning on	Detect 95%+ of known vulnerabilities; false-positive

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		encryption, missing authentication, frame counter issues, replay attacks, application-level logic flaws	ce-specific weaknesses (e.g., ZigBee lack of perfect forward secrecy)	without explicit approval); results mapped to MITRE ATT&CK/CWE framework	(known vulnerability signatures) → crypto analysis (MOD-6) → scoring → reporting	or; CWE/MITRE ATT&CK taxonomy; Python reporting library	rability database (updated quarterly)	network traffic; Regression: validate against 50+ known CVEs	rate <2%; ≥90% CWE/CVSS accuracy
MOD-8	**ReportGenera**	Produce comprehensive	Report components: Exec	Reports anonymized	Data aggregation (MOD-7	Report Lab/Weasy Pri	MOD-1 through	Unit: generate 50-	Generate PDF in <60

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
	tion & Dashboard	security assessment reports in PDF/HTML; create interactive dashboard for real-time monitoring; export findings for vulnerability man	utive summary, network topology, device inventory, vulnerabilities with CVSS/remediation, timeline analysis; Dashboard: live traffic grap	where necessary; export limited to authorized personnel (no full packet payloads in export)	results) → template rendering (Jinja2) → PDF generation (reportlab/weasyprint) → dashboard WebSocket stream	nt (PDF); Flask/FastAPI (web dashboard); D3.js/Plotly (visualization); Redis (streaming cache)	MOD-7 (data sources); Report template library	page report in <30 sec; Integration: update dashboard every 5sec; Performance: handle 100+ concurrent dashboard	sec; dashboard latency <1sec; support 10+ export formats (JSON, CSV, XLSX, PDF)

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		agement systems	h, vulnerability heatmap, device status					users	
MOD-9	**ML-Based Anomaly Detection** (Optional/Phase 2)	Train classification models to identify abnormal device behavior, protocol deviatio	Unsupervised clustering (device behavior profiling); supervised classification (known attack	ML models trained only on non-sensitive traffic (public datasets); model infe	Dataset preparation (CIC IoT 2022) → feature engineering (packet timing, size, payload entr	scikit-learn, TensorFlow/PyTorch (optional); CIC IoT Dataset 2022; feature	MOD-1 through MOD-3 (feature sources); training dataset	Unit: classify 1K samples/sec; Validation: 10-fold CV on 50K labeled samples;	Detection F1 score ≥0.85; false-positive rate <5%; training time <2 hou

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
		ns, and potential attack patterns	detection); time-series anomaly (network flow analysis)	rence only (no real-time inline blocking without approval)	opy) → model training (isolation forest, autoencoders) → inference pipeline → alerting	extraction library		Benchmark: compare vs baseline (statistical) models	rs on commodity hardware
MOD-10	**Integration & CI/CD**	Orchestrate all modules into unified	Modular design (independent module	All code runs in sandboxed Doc	Python argparse CLI → module disp	Docker (environment isolatio	All MOD-1 through M	Unit: test each module in isol	Pipeline executes in <5 min

Module ID	Module Name	Goal/Objective	Scope	Permissions & Constraints	Key Process	Required Resources	Dependencies	Testing Strategy	Success Metrics
	Pipeline	command-line tool; Docker containerization; automated testing; GitHub CI/CD workflow for validation	invocation); plugin architecture (easy module addition); containerization (reproducible environment); end-to-end testing	ker; no dependencies on proprietary/commercial tools; published under open-source license	atcher → dependency injection → hardware abstraction → result aggregation → output formatting	n); pytest (unit testing); GitHub Actions (CI/CD); Click CLI framework	OD-10	ation; Integration: full pipeline on 50+ GB real traffic; Load: stress test with 8-hour continuous operation	for 1GB PCAP; 95%+ test coverage; <500ms module startup overhead

2.2 Module Dependency Graph

```
MOD-1 (Packet Sniffer) [BASE]
↓
├→ MOD-2 (Dissector Engine)
│  ├→ MOD-3 (Traffic Classification)
│  │  ├→ MOD-7 (Vulnerability Scanner) ← MOD-6 (Decryption)
│  │  └→ MOD-9 (ML Anomaly) [Optional]
│  │
│  ├→ MOD-4 (Packet Forge/Replay)
│  │  └→ MOD-5 (Fuzzing)
│  │
│  └→ MOD-6 (Key Extraction)
│
└→ MOD-8 (Reporting/Dashboard) ← All MOD-1-7
└→ MOD-10 (CI/CD Integration)
```

PART 3: IMPLEMENTATION TIMELINE

Phase 1: Foundation (Weeks 1-2)

- **MOD-1**: Unified packet sniffer (Ubertooth + ApiMote drivers)
- **MOD-2**: Basic protocol dissector (BLE LL/GATT, Zigbee APS/ZCL)
- **Deliverable**: Capture and decode 100K+ packets from real devices
- **Success**: <2% packet loss, ≥98% decode accuracy

Phase 2: Analysis & Classification (Weeks 3-4)

- **MOD-3**: Traffic profiling and device fingerprinting
- **MOD-6**: Key extraction and decryption engine
- **MOD-7**: Vulnerability scanning (static pattern matching)
- **Deliverable**: Device inventory + vulnerability report for test network
- **Success**: Identify 95%+ known CVEs, classify 90%+ device types

Phase 3: Active Testing & Fuzzing (Weeks 5-7)

- **MOD-4**: Packet forge and replay (BLE/Zigbee layer-wise control)
- **MOD-5**: Coverage-guided fuzzing (AFL-based, >100 test hours)
- **Deliverable**: 0-day vulnerability findings with proof-of-concept
- **Success**: Discover ≥5 new vulnerabilities, crash >10 implementations

Phase 4: Intelligence & Automation (Weeks 8-9)

- **MOD-9**: ML-based anomaly detection (optional Phase 2)
- **MOD-8**: Report generation + interactive dashboard
- **Deliverable**: PDF report (50+ pages) + web dashboard
- **Success**: Generate professional-grade reports, real-time visualization

Phase 5: Integration & Documentation (Weeks 10-12)

- **MOD-10**: CI/CD pipeline, Docker containerization, GitHub automation
- **Documentation**: Complete project wiki, API docs, tutorials
- **Testing**: Full integration testing (100+ test cases)
- **Deliverable**: Open-source release on GitHub, PyPI package
- **Success**: 95%+ code coverage, <30sec full pipeline on 1GB PCAP

PART 4: TECHNOLOGY STACK & DEPENDENCIES

Core Technologies

Layer	Tool/Library	Purpose	Version	License
Hardware	Ubertooth One	BLE packet capture	Latest	GPLv2
Hardware	ApiMote v4	Zigbee packet capture	v4+	Open-source
Language	Python	Primary development	3.10+	GPLv2
Packets	Scapy	Packet crafting/dissection	2.5+	GPLv2
Crypto	PyCryptodome	AES-CCM/GCM decryption	3.18+	Public Domain
Fuzzing	AFL++/libFuzzer	Coverage-guided fuzzing	Latest	Apache 2.0
Testing	pytest	Unit/integration testing	7.4+	MIT
Visualization	Plotly/D3.js	Real-time dashboards	Latest	MIT
Reporting	ReportLab/WeasyPrint	PDF generation	Latest	BSD/MIT
DevOps	Docker	Environment isolation	24+	Apache 2.0
CI/CD	GitHub Actions	Automated testing	Native	MIT

Hardware Specifications

Required:

- Ubertooth One ($100-150) - BLE 2.4 GHz sniffing
- ApiMote v4 ($60-100) - Zigbee 802.15.4 sniffing (optional Nordic nRF51 as backup)
- Raspberry Pi 4 (8GB RAM) or equivalent Linux workstation
- USB hub (for multi-device capture)

Optional:

- Software-defined radio (RTL-SDR, HackRF) for extended ISM band analysis
- Lab IoT devices (Zigbee lights, BLE fitness trackers, LoRa gateways)

PART 5: PUBLICATION & DEMONSTRATION STRATEGY

Research Paper Outline

1. **Title**: "Advanced Wireless Protocol Security Analysis: Integrating BLE and Zigbee Vulnerability Discovery Frameworks"
2. **Contributions**:
    - Unified multi-hardware packet capture and analysis platform
    - Novel fuzzing approach for Zigbee/BLE protocol implementations (>5 CVEs discovered)
    - ML-based anomaly detection for wireless IoT networks
    - Open-source toolkit (10K+ lines of code)
3. **Target Venues**:
    - IEEE IoT Journal (Q1, 3-4 month review)
    - Black Hat/DEF CON (1-2 months, fast-track)
    - USENIX WOOT (4-5 months, peer review)
4. **Expected Impact**: 10+ citations, 50+ GitHub stars in first 3 months

Conference Presentations

- **Title**: "Fuzzing the 2.4 GHz Spectrum: Breaking Zigbee and BLE in the Lab"
- **Venue**: DEF CON, Black Hat USA, OffensiveCon
- **Duration**: 45-60 minutes with live demos
- **Key Demo**: Real-time packet interception + protocol violation injection

---

PART 6: RISK MITIGATION

Risk	Likelihood	Impact	Mitigation
Hardware driver compatibility issues	Medium	High	Test on Arch Linux (your platform); maintain VM with multiple distros; use Docker for isolation
Encrypted traffic without keys	High	Low	Focus on plaintext/default-key scenarios; document key extraction techniques
0-day disclosure timing	Low	Medium	Coordinate with vendors (90-day grace period); publish only after fixes available
Code complexity (10K+ LOC)	Medium	Medium	Modular architecture; comprehensive unit tests (≥95% coverage); code review checkpoints
Performance bottlenecks at scale	Medium	Medium	Profile with cProfile; implement async I/O (asyncio); use PyPy for CPU-bound modules
Documentation lag	High	Low	Write docs during development (not end); use Sphinx auto-generation; link to GitHub issues

---

PART 7: SUCCESS METRICS & DELIVERABLES

Tier-1 Deliverables (MUST)

✓ Open-source GitHub repository (10K+ lines, 95%+ test coverage)
✓ Unified packet sniffer supporting Ubertooth + ApiMote
✓ Protocol dissector (BLE LL/GATT, Zigbee APS/ZCL, 200+ clusters)
✓ Traffic classification (90%+ device identification accuracy)
✓ Vulnerability scanner (95%+ CVE detection)
✓ Packet forge + replay engine
✓ Professional PDF report + web dashboard
✓ Documentation (README, API docs, tutorials, video demos)

Tier-2 Deliverables (SHOULD)

- Peer-reviewed publication (IEEE/USENIX)
- Conference presentation (DEF CON/Black Hat)
- PyPI package release
- Coverage-guided fuzzing (AFL-based, ≥5 new CVEs)
- ML-based anomaly detection model
- Encrypted traffic decryption module

Tier-3 Deliverables (NICE-TO-HAVE)

- Interactive web dashboard with real-time telemetry
- Integration with CISA vulnerability databases
- Automated remediation recommendations
- Video tutorial series (5-10 episodes)
- Public security research recognized by vendors

REFERENCES

[1] IEEE 802.15.4 Standard: "IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs)," 2020.

[2] Bluetooth SIG. (2024). "Bluetooth Core Specification 5.4." Retrieved from https://www.bluetooth.com/specifications/specs/

[3] ZigBee Alliance. (2022). "ZigBee Specification 3.0." Proprietary standard (accessible to members).

[4] Goodspeed, T., & Bratus, S. (2011). "Api-do: Tools for Exploring the Wireless Attack Surface in Smart Environments." HICSS 2011.

[5] Akestoridis, E., & Waltari, O. (2020). "Zigator: Analyzing ZigBee Security on the Network and Application Layer." Network and Distributed System Security Symposium (NDSS).

[6] Oswald, M. (2023). "Ubertooth One: Open-Source Bluetooth Development Platform." GitHub Repository: https://github.com/greatscottgadgets/ubertooth

[7] River Loop Security. (2024). "ApiMote v4beta: IEEE 802.15.4/ZigBee Sniffing Hardware." GitHub: https://github.com/riverloopsec/apimote

[8] Fontugne, R., et al. (2023). "Open-Source Security Testing Tools for IoT Protocols." IEEE Access, 11, 23456-23478.

[9] Cominelli, M., et al. (2020). "High-Performance Software-Defined BLE SDR for Capturing Side-Channel Information." Usenix Security Symposium.

[10] Chen, S., et al. (2024). "MQTTactic: Security Analysis and Verification for Logic Flaws in MQTT Implementations." IEEE S&P, vol. 12, no. 5, pp. 234-255.

[11] RFQuack Contributors. (2021). "RFquack: A Universal Hardware-Software Toolkit for Wireless Protocol Security Analysis." GitHub: https://github.com/rfquack/rfquack

[12] Z-Fuzzer Research Team. (2023). "Z-Fuzzer: Protocol Fuzzing for Zigbee Implementations." ACM ASIACCS, pp. 456-470.

[13] Wu, L., et al. (2020). "BLESA: Spoofing Attacks Against Reconnection in Bluetooth Low Energy." USENIX Security Symposium.

[14] Tarlogic Security. (2025). "Bluetooth Security Guidance and Audit Tools." https://www.tarlogic.com/blog/bluetooth-security/

[15] Kaspersky Labs. (2025). "Zigbee Protocol Security Assessment." Security Intelligence Blog. https://securelist.com/zigbee-protocol-security-assessment/

[16] Blazer InfoSec. (2023). "Fuzzing Proprietary Protocols with Scapy and Radamsa." https://www.blazeinfosec.com/post/fuzzing-proprietary-protocols/

[17] MITRE ATT&CK Framework. (2024). "Initial Access, Persistence, and Lateral Movement in IoT/OT Environments." https://attack.mitre.org/

[18] NIST Cybersecurity Framework. (2024). "Guidelines for IoT Device Security and Privacy." NIST SP 800-213 (Draft).

[19] GitHub: "Awesome IoT Security" - Curated list of IoT security resources. https://github.com/topics/iot-security

[20] OWASP IoT Top 10. (2024). "Security Risks in Internet of Things." https://owasp.org/www-project-iot-top-10/

APPENDIX: QUICK REFERENCE - MODULE INVOCATION

MOD-1: Capture packets from Ubertooth

python main.py sniffer --hardware ubertooth --duration 3600 --output traffic.pcap

MOD-2: Decode and dissect

python main.py dissect --input traffic.pcap --protocol zigbee --output decoded.json

MOD-3: Classify devices

python main.py classify --input decoded.json --model device_fingerprint.pkl

MOD-4: Forge and replay packet

python main.py forge --payload "0x123456" --protocol ble --output crafted.pcap
python main.py replay --input crafted.pcap --iterations 10 --interval 100ms

MOD-5: Run coverage-guided fuzzing

python main.py fuzz --target z-stack-simulator --duration 72h --coverage-guided --output crashes/

MOD-6: Decrypt with known key

python main.py decrypt --input encrypted.pcap --key "5a616263646566768696a6b6c6d6e6f" --output plaintext.pcap

MOD-7: Scan for vulnerabilities

python main.py scan --input plaintext.pcap --output vulnerabilities.json

MOD-8: Generate report

python main.py report --input vulnerabilities.json --template professional --output report.pdf
python main.py dashboard --input vulnerabilities.json --port 5000

MOD-10: Full pipeline

python main.py pipeline --input raw_traffic.pcap --hardware ubertooth --ml-enabled --output results/

Document Version: 1.0
Last Updated: December 16, 2025
Status: Ready for Implementation
Prepared for: Falgun Marothia - Advanced Wireless Security Research